PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Michael Kollmann	Examiner:	John Romano
Serial No.	10/615,323	Group Art Unit:	2192
Filed:	07/07/2003	Docket No.	CA920030064US1
Customer No.:	65814	Confirmation No.:	9355
Title:	AUTOMATIC COLLECTION OF TRACE DETAIL AND HISTORY DATA		

CERTIFICATE OF TRANSMISSION

I hereby certify that this document is being transmitted electronically to the United States Patent and Trademark Office via the EFS Web e-Filing system on June 7, 2007.

Name: Margo Maddux

AMENDMENT AND RESPONSE TO FINAL OFFICE ACTION

MAIL STOP: AMENDMENT
Commissioner for Patents
Post Office Box 1450
Alexandria, Virginia 22313-1450

Sir/Madam:

In response to the Office Action mailed April 10, 2007, kindly enter the following amendments:

Amendments to the Claims are reflected in the listing of claims which begins on page 2 of this paper.

Remarks begin on page 6 of this paper.

AMENDMENTS TO THE CLAIMS

1 - 25. **(Canceled)**

26. **(New):** A method of automatically adjusting a level of trace data collection, comprising:

monitoring program activity occuring during execution of a computer program;

collecting trace data representative of the program activity;

writing the trace data to one or more trace records, each of the one or more trace records including a trace level associated therewith, the trace level indicating a severity of the program activity;

storing the one or more trace records in a trace history buffer located in volatile memory, such that trace records are written to the trace history buffer until the trace history buffer is full, and older trace records are overwritten by newer trace records when the trace history buffer is full, the trace history buffer thereby containing a history of recent trace records;

comparing, for each trace record stored in the trace history buffer, the trace level to a predetermined threshold value, and writing the trace record to a log file located in persistent storage as a logged trace record if the trace level is greater than the predetermined threshold value;

writing the trace history buffer to the log file if a trap value specific to the program activity is detected within the logged trace record;

writing the trace history buffer to the log file if the trap value specific to the program activity fails to be detected, and if the trace level associated with the logged trace record is greater than a predetermined trace history level;

upon writing the trace history buffer to the log file, resizing the trace history buffer if is determined the trace history buffer is in need of resizing; and

resetting and clearing the trace history buffer such that storing of trace records may continue.

27. **(New):** The method of claim 26 wherein the trace level is a numeric value.

28. **(New):** The method of claim 26 wherein the trace history buffer is of a configurable size.

29. **(New):** The method of claim 26 wherein the predetermined threshold value is configurable.

30. **(New):** The method of claim 26 wherein the predetermined trace history level is configurable.

31. **(New):** The method of claim 26 wherein the predetermined trace history level indicates a level of severity that causes the trace history buffer to be written to the log file.

32. **(New):** The method of claim 26 wherein the trap value comprises a condition code unique to an event occurring within the program.

33. **(New):** The method of claim 26 wherein the trap value comprises a trigger received from a hardware signal.

34. **(New):** The method of claim 26 wherein resizing the trace history buffer further comprises clearing the trace history buffer.

35. **(New):** The method of claim 26 wherein the log file and the trace history buffer reside on different computer systems that communicate over a network.

36. **(New):** A computer program product comprising a computer useable medium having a computer readable program, wherein the computer readable program when executed on a computer causes the computer to:

monitor program activity occuring during execution of a computer program;

collect trace data representative of the program activity;

write the trace data to one or more trace records, each of the one or more trace records including a trace level associated therewith, the trace level indicating a severity of the program activity;

store the one or more trace records in a trace history buffer located in volatile memory, such that trace records are written to the trace history buffer until the trace history buffer is full, and older trace records are overwritten by newer trace records when the trace history buffer is full, the trace history buffer thereby containing a history of recent trace records;

compare, for each trace record stored in the trace history buffer, the trace level to a predetermined threshold value, and writing the trace record to a log file located in persistent storage as a logged trace record if the trace level is greater than the predetermined threshold value;

write the trace history buffer to the log file if a trap value specific to the program activity is detected within the logged trace record;

write the trace history buffer to the log file if the trap value specific to the program activity fails to be detected, and if the trace level associated with the logged trace record is greater than a predetermined trace history level;

upon writing the trace history buffer to the log file, resize the trace history buffer if is determined the trace history buffer is in need of resizing; and

reset and clear the trace history buffer such that storing of trace records may continue.

37. **(New):** The computer program product of claim 36 wherein the trace level is numeric.

38. **(New)**: The computer program product of claim 36 wherein the trace history buffer is of a configurable size.

39. **(New):** The computer program product of claim 36 wherein the predetermined threshold value is configurable.

40. **(New):** The computer program product of claim 36 wherein the predetermined trace history level is configurable.

41. **(New):** The computer program product of claim 36 wherein the predetermined trace history level indicates a level of severity that causes the trace history buffer to be written to the log file.

42. **(New):** The computer program product of claim 36 wherein the trap value comprises a condition code unique to an event occurring within the program.

43. **(New):** The computer program product of claim 36 wherein the trap value comprises a trigger received from a hardware signal.

44. **(New):** The computer program product of claim 36 wherein resizing the trace history buffer further comprises clearing the trace history buffer.

REMARKS

The office action issued by the Examiner dated April 10, 2007 and the citations referred to in the office action have been carefully considered. In response, claims 1, 3, 4, 7-9, 11-13, 16-18, 20, 21, 24, and 25 are cancelled without prejudice or disclaimer. Claims 26–44 are newly added with this response. No new matter is added with the introduction of these new claims. Claims 26-44 are now pending in the application.

The Applicant thanks the Examiner for the courtesy extended during the telephone interview conducted on May 25, 2007.

Claim Rejections – 35 USC § 112

Claims 1, 3, 4, 7-9, 11-13, 16-18, 20, 21, 24, and 25 were rejected under 35 U.S.C. § 112, second paragraph as being indefinite. Claims 1, 3, 4, 7-9, 11-13, 16-18, 20, 21, 24, and 25 are therefore cancelled without prejudice or disclaimer. Newly added claims 26-44 are introduced to more particularly point out and distinctly claim the subject matter of the present application. Applicant submits that all of the rejections under 35 U.S.C. 112 as set forth in the office action have been overcome through the cancellation of such rejected claims, and introduction of newly added claims 26-42.

Claim Rejections – 35 U.S.C. § 103

Claims 1, 3, 4, 7-9, 11-13, 16-18, 20, 21, 24, and 25 were rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent Publication No. 2002/0198983 to Ullman et al. in view of US Patent No. 7,058,928 to Wygodny et al. and further in view of US Patent No. 5,642,478 to Chen. As mentioned above, claims 1, 3, 4, 7-9, 11-13, 16-18, 20, 21, 24, and 25 are cancelled without prejudice or disclaimer.

Ullman discloses a system and method of tracking program activities in a distributed computing environment. Ullman is directed to a centralized logging utility for tracing and

logging a plurality of programs/applications on multiple clients and/or servers. (e.g. see Ullman, Abstract, par. [0019], Fig. 1) This is accomplished by monitoring high level system messages and detecting for errors. (e.g. see Ullman pars. [0021] – [0023], Fig. 2) When an error is detected in a high level system message, the source location of the error is determined such that tracing and logging can be selectively initiated at the source location. (e.g. see Ullman pars. [0023] – [0025], Fig. 6)

As recited in newly added independent claim 26, a method of automatically adjusting the level of trace data collection is claimed. Trace data is collected and written to one or more trace records. The trace records are stored in a trace history buffer. The trace history buffer is located in volatile memory, such that trace records are written to the trace history buffer until the trace history buffer is full, and older trace records are overwritten by newer trace records when the trace history buffer is full. The trace history buffer thereby contains a history of the most recent trace records.

Next, an analysis is conducted to determine if one or more trace records should be written to a log file. That is, independent claims 26 and 36 recite "comparing, for each trace record stored in the trace history buffer, the trace level to a predetermined threshold value, and writing the trace record to a log file located in persistent storage as a logged trace record if the trace level is greater thatn the predetermined threshold value". Independent claims 26 and 36 also recite "writing the trace history buffer to the log file if a trap value specific to the program activity is detected within the logged trace record". Even further, independent claims 26 and 36 recite "writing the trace history buffer to the log file if the trap value specific to the program activity fails to be detected, and the trace level associated with the logged trace record is greater than a predetermined trace history level".

Therefore, a multi-tiered analysis is performed to determine whether or not trace records should be written to a log file. First, a determination of whether each individual trace record should be written to the log file occurs. If it is determined the trace record should be written to the log file, a second determination is made regarding whether the historical trace records stored

in the trace history buffer should also be written to the log file. The second determination of whether the historical trace data should also be written to the log file is further based on detecting a trap value or detecting if the trace level is greater than a predetermined trace history level.

However, Ullman does not disclose such a multi-tiered analysis in determining whether and how trace records should be written to a log file. While the Examiner cites Ullman's disclosure of "a filter (that) determines which types of message and trace records are to be processed by the logger" (p.2, par. [0022]), there is no disclosure in Ullman of any secondary determination for whether the history contained in the trace history buffer should additionally be written to the log file.

Ullman does not even disclose writing the trace records to a trace history buffer located in non-persistent or volatile memory, thereby keeping a history of the most recent trace records, such that that one or more trace records can selectively be written to a log file in persistent storage. Ullman only discloses writing log messages to some type of persistent storage 238, and there is no disclosure in Ullman of storing trace data to a non-persistent/volatile memory.

Furthermore, Ullman does not teach or disclose writing the trace history buffer to the log file. Ullman teaches selectively initiating tracing and logging starting after the detection of a specified event. The present application is directed to continuously tracing, and selectively logging trace data and historical trace data occuring prior to and including a specific activity/event.

On page 8 of the office action, the Examiner likens Ullman's disclosure of dumping the stack to writing the trace history buffer to the log file as is claimed. However this is not an accurate interpretation. Ullman states:

> Given the nature of an exception message, which may originate at one component ... and then be propogated back to other components ... , a plurality of exception messages may be provided ... from a plurality of components/location, when in fact only one

> exception event has occurred." Therefore, it is most advantageous upon receipt of an exception message to utilize existing system resources, specifically the "dump stack", to **trace back to component at which the task started**. See Ullman, p. 4, par. 0035, empasis added.

It is clear from Ullman's description that exceptions may arise at one component and propogate back to other components. Therefore, the location of the component at which the task relating to the exception started needs to be identified in order to initiate tracing and logging. The dump stack is simply used to identify which specific component or location, from a plurality of components/locations, where the exception started. This teaching is consistent with Ullman being directed to tracking program activities in a distributed computing environment having multiple computers and/or components/programs. Ullman's use of the dump stack has nothing to do with writing trace history data to a log file, as is claimed.

Therefore, as described above, Ullman does not disclose or teach the limitations as recited in independent claims 26 and 36. As a result, the combination of Ullman with Wygodny and Chen does not teach claims 26 or 36. Accordingly, Applicant submits that claims 26 and 36 are in condition for allowance. Further, Applicant submits that claims 27-35 and 37-44 are allowable as these claims depend from claims 26 and 36, respectively.

Therefore, because all the claims limitations are not taught or suggested by Ullman et al. alone or in combination with Wygodny et al. and Chen, these references cannot be used as the basis of a rejection under 35 U.S.C. §103(a). Further, if an independent claim is non-obvious under 35 U.S.C. 103, then any claim depending there from is non-obvious. *In re Fine*, 837 F.2d 1071, 5 U.S.P.Q.2d 1596 (Fed. Cir. 1988). In light of these amendments and arguments, Applicant has overcome the Examiner's 35 U.S.C. §103(a) rejections. Thus, it is respectfully submitted that all of the Examiner's objections have been successfully traversed and that the application is now in order for allowance. Accordingly, reconsideration of the application and allowance thereof is courteously solicited.

If, for any reason, the Examiner finds the application other than in condition for allowance, the Examiner is respectfully requested to call Applicant's undersigned representative,

Margo Maddux at **(310) 584-4249** to discuss the steps necessary for placing the application in a condition for allowance.

The Director is authorized to charge any additional fee(s) or any underpayment of fee(s), or to credit any overpayments to **Deposit Account Number 09-0460**. Please ensure that Attorney Docket Number CA920030064US1 is referred to when charging any payments or credits for this case.

Respectfully submitted,

Date: June 7, 2007

Margo Maddux

Margo Maddux, Patent Agent
Reg. No. 50,962

Customer Number 65814
PATENT INGENUITY, P.C.
520 Broadway, Suite 350
Santa Monica, CA 90401
Phone: (310) 584-4249
Fax: (310) 564-0454
E-mail: patents@patentingenuity.com